Physics-based Seismic Hazard Analysis on Petascale Heterogeneous Supercomputers

Y. Cui[1], E. Poyraz[1], K. B. Olsen[2], J. Zhou[1], K. Withers[2], S. Callaghan[3], J. Larkin[4], C. Guest[1], D. Choi[1], A. Chourasia[1], Z. Shi[2], S. M. Day[2], P. J. Maechling[3], T. H. Jordan[3]

[1]University of California, San Diego, [2]San Diego State University, [3]University of Southern California, [4]NVIDIA Inc.

[1](yfcui,epoyraz,j4zhou,cguest,djchoi,amit@ucsd.edu),
[2](kbolsen,withers,zshi,sday@mail.sdsu.edu),
[3](scottcal,maechlin,tjordan@usc.edu), [4](jlarkin@nvidia.com)

ABSTRACT

We have developed a highly scalable and efficient GPU-based finite-difference code (AWP) for earthquake simulation that implements high throughput, memory locality, communication reduction and communication / computation overlap and achieves linear scalability on Cray XK7 Titan at ORNL and NCSA's Blue Waters system. We simulate realistic 0-10 Hz earthquake ground motions relevant to building engineering design using high-performance AWP. Moreover, we show that AWP provides a speedup by a factor of 110 in key strain tensor calculations critical to probabilistic seismic hazard analysis (PSHA). These performance improvements to critical scientific application software, coupled with improved co-scheduling capabilities of our workflow-managed systems, make a statewide hazard model a goal reachable with existing supercomputers. The performance improvements of GPU-based AWP are expected to save millions of core-hours over the next few years as physics-based seismic hazard analysis is developed using heterogeneous petascale supercomputers.

Keywords

SCEC, seismic hazard analysis, earthquake ground motions, weak scaling, GPU, CyberShake, hybrid heterogeneous.

1. INTRODUCTION

Economic exposure to earthquake devastation is skyrocketing, primarily because urban environments are growing rapidly in seismically active regions. Probabilistic seismic hazard analysis (PSHA) in its standard form—i.e. derived empirically assuming the hazard is time-independent (Poissonian)—has been effective in helping decision-makers reduce seismic risk and increase community resilience. However, the earthquake threat is highly time-dependent and involves terribly violent, but known, physics.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
SC13 November 17-21, 2013, Denver, CO, USA
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-2378-9/13/11...$15.00.
http://dx.doi.org/10.1145/2503210.2503300

To understand risk and improve resilience, we need to quantify earthquake hazards in physics-based models that can be coupled to engineering models of the built environment [19]. For example, the performance of critical distributed systems (water, medical, energy, etc.) depends on how complex earthquake wavefields interact nonlinearly with the mechanical heterogeneities of an entire cityscape, below as well as above the ground. Earthquake system science seeks to represent these complexities and interactions in system-level models.

The enabling technology of earthquake system science and physics-based PSHA is the numerical simulation of fault rupture dynamics and seismic wave propagation in realistic 3D models of the crust's heterogeneous structure. The Southern California Earthquake Center (SCEC) established its "Community Modeling Environment" in 2001 as a collaboratory for earthquake simulation. In 2005, we demonstrated the jump to terascale with the TeraShake simulations at the San Diego Supercomputer Center [22, 23]. We discovered how the rupture directivity of the southern San Andreas fault, a source effect, could couple to the excitation of sedimentary basins, a site effect, to substantially increase the seismic hazard in Los Angeles [22]. We also used dynamic rupture simulations to investigate how increases in source complexity can act the other way, reducing the directivity effect [23].

In 2008, the USGS adopted a SCEC simulation of a magnitude-7.8 San Andreas earthquake [16] as the basis for the first Great Southern California ShakeOut, the largest earthquake preparedness exercise of its time. This simulation was verified by comparisons of simulations from three different codes (Graves, Hercules, AWP-ODC) running at three national supercomputer centers [1]. The ShakeOut simulation provided system engineers, emergency responders, and disaster planners with a realistic, high-resolution scenario, prompting many detailed studies of a Katrina-scale catastrophe that have led to reductions in risk and improved resilience [24]. ShakeOut exercises are now performed in most of the United States and a growing number of other countries [27].

Improvements in earthquake simulation have closely tracked the development of leadership-class computational facilities. A major goal of the SCEC program—to simulate the largest expected ("wall-to-wall") earthquake on the San Andreas fault up to seismic frequencies exceeding 1 Hz—was achieved in 2010 on *Jaguar*, the first petascale machine at ORNL's Oak Ridge Leadership Computing Facility (OLCF). This Magnitude-8 (M8) scenario involved running the AWP-ODC finite-difference code on a uniform mesh comprising 436-billion elements for 24 hours

at a sustained 220 Tflops [7]. The computational size of the M8 simulation (mesh points × time steps) was almost 10^{17}, more than three orders of magnitude greater than the TeraShake simulations. M8 highlighted the importance of two aspects of earthquake physics in hazard analysis: the propagation of dynamic fault ruptures at speeds exceeding the shear-wave speed, "super-shear rupturing", which substantially modifies the seismic wavefield, and the existence of stresses high enough in the near-source and near-surface regions to require a nonlinear treatment of the deformations.

The arrival of petascale computing has opened the door to full-scale, physics-based PSHA. For example, earthquake simulations have recently been validated against ground motions recorded up to 4 Hz, with promising results [28], and we are pushing these comparisons to even higher frequencies. However, in order to calculate seismic hazards in California and other tectonically active regions, simulating just a few earthquakes will not do; we must adequately sample earthquake distributions from probabilistic models, such as the Uniform California Earthquake Rupture Forecast (UCERF) [11]. Using standard "forward" simulation methods, computing three-component seismograms from M sources at N sites requires M simulations. For the UCERF2 model in Southern California, $M > 10^5$; i.e., hundreds of thousands to millions of possible earthquake sources must be modeled, which cannot be done directly, even at petascale.

To overcome this scale limitation, SCEC has built a special simulation platform, CyberShake, which uses the time-reversal physics of seismic reciprocity to turn the problem around [17]. A complete tensor-valued wavefield (the strain Green tensor or SGT) is calculated for a system of point forces at surface sites; seismic reciprocity then allows us to compute seismograms at those sites by fast quadratures of the SGT over the fault surfaces. This "reciprocal" simulation method can generate 3-component seismograms for M sources at N sites with only $3N$ simulations. For the Los Angeles region, the near-surface geologic structure can be interpolated to produce high-resolution seismic hazard maps with N as small as 200-250, reducing the computations by a factor of 2,000. Scientific workflow software is used to manage the hundreds of millions of jobs needed to populate a CyberShake model [3].

Using the CyberShake platform, we have created the first physics-based PSHA models of the Los Angeles region from suites of simulations comprising ~10^8 seismograms. These models are "layered", allowing earthquake engineers and other users to access ensembles of hazard curves (representing epistemic uncertainties), to disaggregate the calculations and identify the ruptures that dominate the hazard at a particular site, and to retrieve the actual seismograms, which can then be used to drive full-physics engineering models (Figure 1).

CyberShake brings the computational challenges of physics-based PSHA into sharp focus. The current models are limited to low seismic frequencies (≤ 0.5 Hz). Goals are to increase this limit to above 1 Hz and produce a California-wide CyberShake model using the new UCERF3 rupture forecast, which is scheduled to be released this year. The computational size of the statewide model will be more than 100 times larger than the current Los Angeles models. Our progress toward exascale is also driven by the application of full-3D waveform tomography to the development of the seismic velocity models [6, 29], which are required as input to CyberShake. Full-3D tomography using 3-component seismograms from M sources observed at N stations requires at least $3N + M$ wavefield simulations per iteration [5].

This paper demonstrates that the computational power needed to accelerate CyberShake, and other applications of earthquake simulations will likely come from accelerator-based, many-core architectures. We present initial results on the performance of a finite-difference code, Anelastic Wave Propagation by Olsen, Day, and Cui (AWP-ODC) [7], that has been GPU-accelerated on the new OLCF system, *Titan*. A hardware-oriented design has been developed to achieve high performance. The main problem is to make full use of GPU power by overlapping slow data communication with an extended computing region. The algorithms used are highly scalable because they carefully arrange the order of computation and communication to hide latency, resulting in speedup and parallel efficiency. We implement a communication model that reduces the intra-node frequency of data movement between CPU and GPU and enables complete overlap of communication and computation. This model can be extended to general stencil computing on a structured grid. We have validated the simulation outputs in comparison to both reference models and earthquake observations.

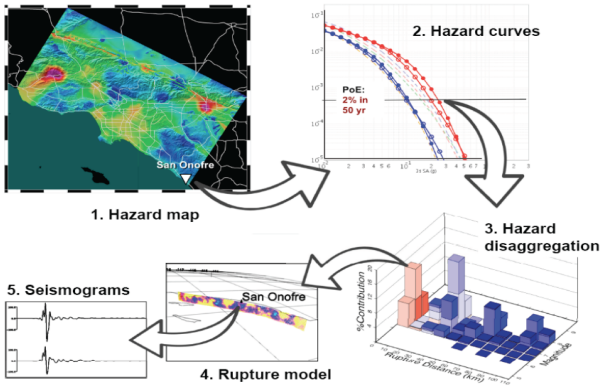

Figure 1: The CyberShake hazard model, showing the layering of information. (1) Hazard map for the LA region (hot colors are high hazard). (2) Hazard curves for a site near the San Onofre Nuclear Generating Station. (3) Disaggregation of hazard in terms of magnitude and distance. (4) Rupture with the highest hazard at the site (a nearby offshore fault). (5) Seismograms simulated for this rupture. Arrows show how users can query the model starting at high levels (e.g. hazard map) to access information of progressively lower levels (e.g. seismograms).

Section 2 of this paper outlines our solution for AWP-ODC, which is based on an effective memory reduction scheme. Section 3 introduces single-GPU and multi-GPU implementation of the solver. Section 4 demonstrates the parallel efficiency and summarizes the sustained performance achieved on petascale supercomputers. Section 5 applies these new capabilities to obtain two scientific results: (1) the first 10-Hz simulation on the *Titan* system and (2) the first CyberShake hazard curve generated using GPU-based AWP-ODC on the NCSA *Blue Waters* system.

2. NUMERICAL METHODS

GPU-accelerated AWP-ODC (hereafter abbreviated as AWP) is based on the FD code originally developed by Kim Bak Olsen at University of Utah [21]. The code solves the 3D velocity-stress wave equations with the explicit staggered-grid FD scheme. This scheme is fourth-order accurate in space and second-order accurate in time. This application has two computing modes: dynamic rupture and wave propagation mode. In this paper, we will focus on the wave propagation mode.

2.1 Wave Propagation Equations

The seismic forward problem calculates the propagation of seismic waves through spatially heterogeneous soil materials. AWP solves a coupled system of partial differential equations. The governing elastodynamic equations can be written as

$$\partial_t v = \frac{1}{\rho}\nabla \cdot \sigma \quad (1a)$$

$$\partial_t \sigma = \lambda(\nabla \cdot v)I + \mu(\nabla v + \nabla v^T) \quad (1b)$$

where λ and μ are the Lamé coefficients, ρ is the density, and v and σ are the particle velocity vector and symmetric stress tensor respectively. Decomposing (1a) component-wise leads to three scalar-valued equations for the velocity vector components and six scalar-valued equations for the stress tensor components.

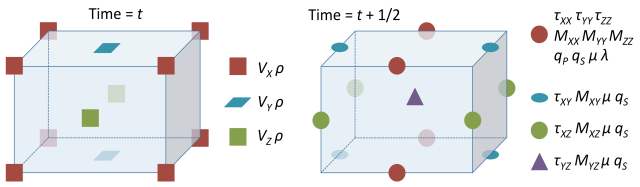

Figure 2: Staggering of the wavefield parameters, where V_i are particle velocities, τ_{ij} are stress tensor components, M_{ij} are memory variables, ρ, λ, μ are elastic parameters, q_p and q_s are quality factors for *P* and *S* waves, respectively.

2.2 Staggered-Grid Finite Difference Equations

The nine governing scalar equations are approximated by finite differences on a staggered grid in both time and space (Figure 2). Time derivatives are approximated by

$$\partial_t v(t) \approx \frac{v\left(t+\frac{\Delta t}{2}\right)-v\left(t-\frac{\Delta t}{2}\right)}{\Delta t} \quad (2a)$$

$$\partial_t \sigma\left(t+\frac{\Delta t}{2}\right) \approx \frac{\sigma(t+\Delta t)-\sigma(t)}{\Delta t} \quad (2b)$$

For the spatial derivatives, let Φ denote a generic velocity or stress component, and h be the equidistant mesh size. The FD approximation to the partial derivative with respect to x at grid point (i,j,k) is

$$\partial_x \Phi_{i,j,k} \approx D_x^4(\Phi)_{i,j,k} = \frac{c_1\left(\Phi_{i+\frac{1}{2},j,k}-\Phi_{i-\frac{1}{2},j,k}\right)+c_2\left(\Phi_{i+\frac{3}{2},j,k}-\Phi_{i-\frac{3}{2},j,k}\right)}{h} \quad (3)$$

with $c_1 = 9/8$ and $c_2 = -1/24$. This equation is used to approximate spatial derivatives for each velocity and stress component.

Truncation of the 3D modeling domain on a computational mesh inevitably generates undesirable reflections. Absorbing boundary conditions (ABCs) are designed and optimized to reduce these reflections to the level of numerical noise. AWP implements ABCs based on simple 'sponge layers' [4]. The ABCs apply a damping term to the full wavefield inside the sponge layer and are unconditionally stable.

2.3 Anelastic attenuation in AWP-ODC

Seismic waves are subjected to anelastic losses in the Earth, and such attenuation must be included in realistic simulations of wave propagation. Anelastic attenuation can be quantified by quality factors for S waves (Q_s) and P waves (Q_p). Early implementations of attenuation models include Maxwell solids (e.g., [14]) and standard linear solid models (e.g., [2]). Here, we implemented an efficient coarse-grained methodology in AWP [9, 10], which significantly improves the accuracy of the stress relaxation schemes. This method closely approximates frequency-independent Q by incorporating a large number of relaxation times (eight in our calculations) into the relaxation function without sacrificing computational performance or memory. The quality factor Q (separate for S and P waves) in this formulation is expressed as

$$Q^{-1}(\omega) \approx \frac{\delta M}{M_u}\sum_{l=1}^{N}\frac{\lambda_l \omega \tau_l}{\omega^2\tau_l^2+1} \quad (4)$$

where δM is the relaxation of the modulus, M_u is the unrelaxed modulus, λ_l are weights used in the associated quadrature calculations, τ_l are the relaxation times, and ω is angular frequency. Each stress component has associated with it N memory variables $\varsigma_l(t)$ (one variable co-located with each stress component in the staggered grid, see Figure 2).

$$\sigma(t) = M_u\left[\varepsilon(t) - \sum_{l=1}^{N}\varsigma_l(t)\right] \quad (5)$$

where $\sigma(t)$ is stress and $\varepsilon(t)$ is strain. We use N=8 to obtain accuracy in the implementation.

2.4 Strain Green Tensor Calculations

Alternatively, the strain Green tensor (SGT) can be simulated and utilized in reciprocal methods to produce waveforms. The strain Green tensor can be calculated as

$$\boldsymbol{H}(\boldsymbol{r},t;\boldsymbol{r}_s) = \frac{1}{2}[\partial_i^S G_{jn}(\boldsymbol{r},t;\boldsymbol{r}_s) + \partial_j^S G_{in}(\boldsymbol{r},t;\boldsymbol{r}_s)] \quad (6)$$

where G_{in} is the ith component of the displacement response to the nth component of a point force at $\boldsymbol{r}_S$, and the spatial gradient operator acts on the field coordinate $\boldsymbol{r}$ [32]. The SGT can be computed from the stress-field by applying the stress-strain constitutive relation. The displacement field is linearly related to the seismic moment tensor $\boldsymbol{M}$:

$$u_n(\boldsymbol{r},t;\boldsymbol{r}_s) = \boldsymbol{H}(\boldsymbol{r}_S,t;\boldsymbol{r}):\boldsymbol{M} \quad (7)$$

Therefore, the elements of the SGT can be used in earthquake source parameter inversions to obtain the partial derivatives of the seismograms with respect to the moment tensor elements. By directly using the strain Green tensor, we can improve the computational efficiency in waveform modeling while eliminating the possible errors from numerical differentiation [32]. Seismic reciprocity can then be applied to compute synthetic seismograms from SGTs, from which peak spectral acceleration values are computed and combined into hazard curves [31, 32].

3. AWP IMPLEMENTATION DETAILS

In this section we present implementation details of our GPU application AWP. We introduce a C/CUDA/MPI implementation whose initial development was part of co-author Zhou's graduate research [33, 34]. We will emphasize the key points that led to the scaling performance we obtained for the GPU application. The code has been verified against other FE and FD codes used in the community, and validated against observed data.

3.1 Stencil Computation Kernel

In AWP, two computation kernels for velocity and stress are carried out in sequence for wave propagation simulations based on the numerical approximation of the partial differential equations (1-3). At each time step in the main loop, for each mesh point in the domain, first the velocity computation kernel updates three velocity components (in *X, Y,* and *Z* directions) by using the six stress components (on *XX, YY, ZZ, XY, XZ,* and *YZ* faces). Then the stress computation kernel employs these updated velocity components to update the six stress components. There

are twenty-one 3D arrays to be maintained in memory to process the wave propagation, including velocity stress variables and coefficients. The size of each 3D array is the same as the 3D simulation domain. Figure 3 shows three examples for the memory access pattern for velocity v_x, stress xx and stress xy computation kernels. Approximately 136 reads, 15 writes and 307 Flops are involved for each point of the 3D domain in one iteration. The Flops to bytes ratio is around 0.5, with low computational intensity. Note this ratio does not include data sharing from one cell to the next, and the performance is strongly dominated by the memory requirements. Improving the data locality has been key to achieving high performance.

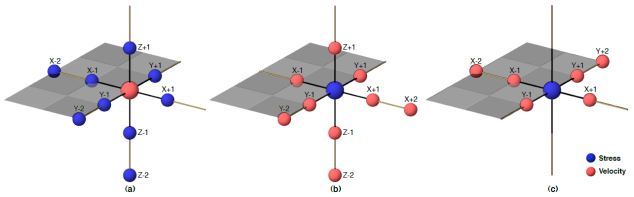

Figure 3: (a) 13-point asymmetric stencil computation for velocity v_x: a velocity center point computation requires 13 points stress input including 4 from the same center location and 9 others from neighborhood. Computation for velocity v_y and v_z has similar format with different neighborhood input. (b) 13-point asymmetric stencil computation for stress (*xx, yy, zz*) with similar stencil format as velocity v_x. (c) 9-point asymmetric stencil computation for stress *xy*: input velocity involves *x* and *y* directions. Computation for stress *yz* and *xz* has similar format in different directions.

3.2 AWP-ODC Fortran/MPI Code

The CPU-based AWP-ODC software is highly scalable, and composed of solvers (dynamic rupture and wave propagation), pre-processing tools (PetaSrcP, PetaMeshP) and other post-processing workflow tools [7]. Scalable IO in the code uses MPI-IO to handle petabytes of simulation data [7]. The code has been verified against other FE and FD codes used in the community [1], and validated against observed data [7, 8, 22, 23]. This simulation software is used as a basis and reference for our GPU-based solver AWP in development and tests.

3.3 Single-GPU AWP Implementation

The AWP code was re-designed to maximize throughput. The initial programming effort was to convert the Fortran/MPI code to a serial CPU program in C. Then CUDA calls and kernels were added to the application for GPU computation [33]. Each GPU is controlled by an associated CPU. The design is implemented with maximum throughput for heterogeneous computing environments in mind.

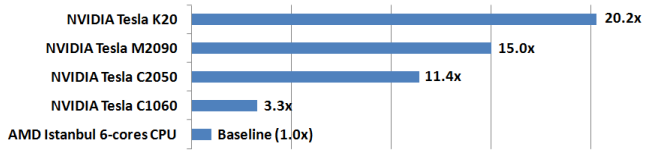

Figure 4: Speedup of single-GPU performance on different NVIDIA architectures. The baseline time is from the Fortran/MPI based CPU code described in Section 3.2 running on AMD Istanbul 6-core CPU at 2.6 GHz with 6 MPI threads for a 3D grid size of 224×224×1024.

Various optimization approaches are implemented to improve the data locality: (1) memory is coalesced for continuous CUDA thread data access, (2) register usage is optimized to reduce global memory access, (3) L1 cache or shared memory usage is optimized for data reuse and register savings, and (4) read-only memory is employed to store constant coefficient variables because of read-only cache benefits [33]. Figure 4 shows approximately 6x performance gain on the latest Kepler architecture compared to NVIDIA's first generation GPU.

3.4 Multi-GPU AWP Implementation

The concept of CUDA/MPI implementation was discussed in [34]. We address some technical details here for completeness, with emphasis on algorithm-level communication reduction, and overlap of communication and computation. The new components added to the code include a two-layer IO model for scalable solution, and an API for high throughput.

3.4.1 Communication and Computation

Unlike the CPU code with 3D decomposition, AWP uses a two layer decomposition where each layer is 2D (Figure 5). The 3D domain (*NX, NY, NZ*) is partitioned into (*PX, PY*, 1) sub-domains. Each GPU is responsible for the computation of its own sub-domain with dimension of (*nx=NX/PX, ny=NY/PY, NZ*). The sub-domain is further partitioned along *Y* and *Z* axes inside the GPU for different streaming multiprocessors (SM).

Benefits of using a 2D decomposition include higher memory locality, reduced memory access latency, and reduced MPI communications. In this fashion, two consecutive locations in the memory correspond to data for two neighboring mesh points in the *Z* direction, full *Z* axis is available for data access and computation. Also the number of neighbors per sub-domain is reduced from 6 to 4 for inner sub-domains, that means no neighboring sub-domains along the *Z* direction. With this approach, the total number of MPI communications is reduced by approximately 33% [34].

The 2D decomposition may introduce increase in volume of communications; however, 2D decomposition reduces the total number of communications, favoring fewer and larger messages. The typical simulation model extent in our California research area has smaller vertical (*Z*) dimension than *X* and *Y* (with a 8×4×1 ratio in 3D). The speed of computation inside a GPU is sensitive to the sub-domain dimensions that we choose, it is more efficient to use larger *Z* dimension in sub-domain dimensions as *Z* is the fastest dimension for memory access in our implementation.

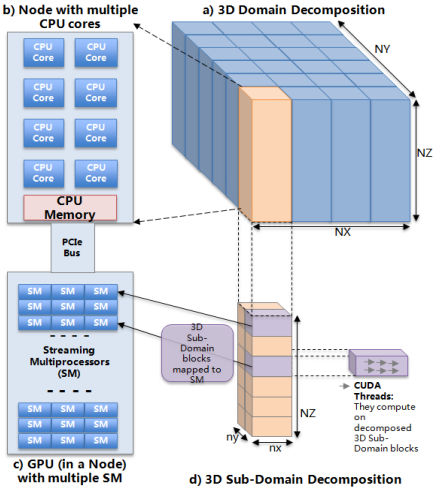

Figure 5: Two-layer 3D domain decomposition: *X&Y* decomposition for GPUs and *Y&Z* for GPU SMs.

To reduce the amount of communication and latency, we extend the ghost cell region by adding two additional layers, and hence manage a ghost cell region with thickness of 4 mesh points per boundary, in both *X* and *Y* directions. We exchange 4 layers of velocity data of ghost cells, resulting in up-to-date velocity data for (*nx*+8, *ny*+8, *NZ*). Then each GPU computes stress for a domain of size (*nx*+4, *ny*+4, *NZ*) including 2 layers of ghost cells in each boundary. The stress is then used to compute velocity for the sub-domain of size (*nx, ny, NZ*) (Figure 6). That means we exchange twice as much velocity data with neighbors without further stress data updates resulting in 33% less data exchange with halved communication frequency, as velocity has three variables and stress has six. This is a significant saving in communication compared to the common model that updates wavefield with neighbors twice per iteration, after both velocity and stress computations, as used in our CPU code [34]. Moreover, this allows more time to overlap communication with computation without synchronizing stress for ghost cell data. The slight increase in memory and computation requirements is bounded above by $NZ \times (4 \times (nx + ny) + 16)$, which can easily be obtained by setting sub-domain size as (*nx*+4, *ny*+4, *NZ*). For our benchmark block size of 160 × 160 × 2048, this upper bound corresponds to a 5% increase in the memory requirement.

The communication approach introduced in Figure 6 requires two extra layers of ghost cells, for we need data for all four corners. We introduced an in-order communication method (first west/east, then north/south) in order not to have additional MPI messages for the diagonal cell information exchange [34].

We employ an ordered scheduling to manage asynchronous communication and computation efficiently (Figure 7). GPU first computes the velocity of the boundary region, which corresponds to ghost cells of a neighboring sub-volume (*V1*-*V4*). While this data is asynchronously copied to CPU and sent to neighbors through MPI, GPU computes the velocity (*V5*) and stress (*S5*) for the inner region. When the data exchange is done and velocity data for the ghost cells is received, it is copied back to GPU asynchronously. After the velocity for ghost cells is copied, GPU computes stress for the boundary region (*S1*-*S4*) [34].

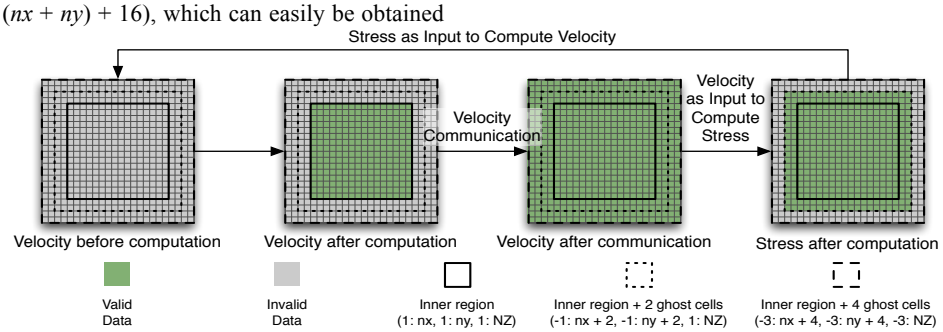
Stress as Input to Compute Velocity
Velocity Communication
Velocity as Input to Compute Stress
Velocity before computation
Velocity after computation
Velocity after communication
Stress after computation
Valid Data
Invalid Data
Inner region (1: nx, 1: ny, 1: NZ)
Inner region + 2 ghost cells (-1: nx + 2, -1: ny + 2, 1: NZ)
Inner region + 4 ghost cells (-3: nx + 4, -3: ny + 4, -3: NZ)

Figure 6: Communication reduction - extend ghost cell region with 2 extra layers and utilize computation instead of communication to update the ghost cell region before stress computation. The 2D *XY* plane represents 3D sub-domain, no communication is required in *Z* direction due to 2D decomposition for GPUs.

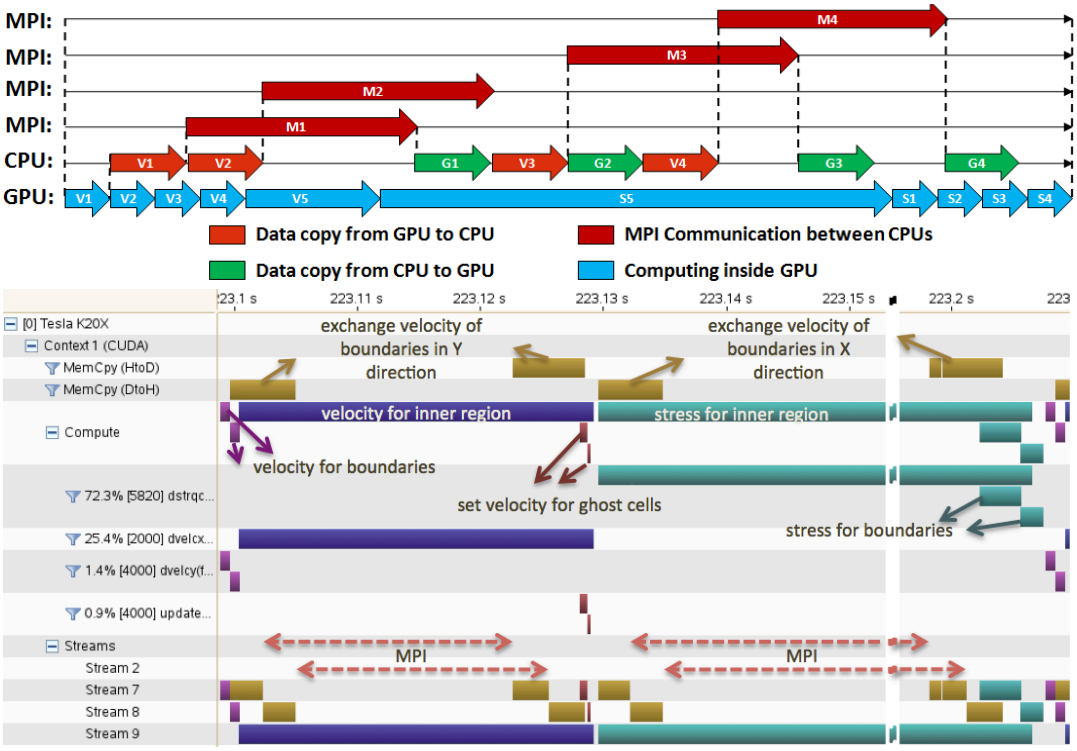

Figure 7: Overlap of computation and communication. Top: concept scheme. Bottom: *nvvp* profiler output. Full overlap is achieved.

3.4.2 I/O

AWP is capable of reading in a large number of dynamic sources and petabytes of heterogeneous velocity mesh inputs. The dynamic sources consist of the positions of earthquake source stations, and stress data associated with each source station. In our 10-Hz simulation case (Section 5), the mesh input is 4.9 TB, and the source is as large as 1.9 TB. These dynamic sources are computed based on the accurate and verified staggered grid, split-node scheme [8]. Millions of sources may be highly clustered in a concentrated grid area, resulting in hundreds of gigabytes of source data assigned to a single core. Copying this data to GPUs through PCIe is an additional challenge at runtime.

We support 3 different modes for reading the sources and the mesh: serial reading of a single file, concurrent reading of pre-partitioned files, and concurrent reading through MPI-IO. Source partitioning involves both spatial and temporal locality required to fit in the GPU memory. Parameters are introduced to control how often the partitioned source is copied from CPUs to GPUs. This feature allows CPUs to read in large chunks of source data to avoid frequent access to file system, while GPU only copies over the amount it can afford. Our implementation has demonstrated linear scalability in handling the initial dataset.

AWP uses MPI-IO to write the simulation outputs to a single file concurrently. This works particularly well as more memory is available on CPUs to allow effective aggregation of outputs in CPU memory buffers before being flushed. We support run-time parameters to select a subset of the data by skipping mesh points as needed.

3.4.3 AWP API Implementation

We also implemented a generic application programming interface (API), which is used in AWP and can be useful to other programs. The API employs *pthreads* to take advantage of the idle CPU cores which can work on other independent tasks in parallel. For each computing node with multiple CPU cores, only 1 core/thread is required to run the regular GPU solver since each node has only 1 GPU on Cray XK7. Hence the other cores are available during the running period. The API allows other workloads to run simultaneously with the GPU solver.

Our first pthread task is the "output" thread. We separated the output related operations from the computation code and implemented them in the output thread. After the main thread finishes the initialization, the output thread starts and passively waits until the main thread signals that it is time to save velocity data. Then the output thread wakes up, launches a kernel on the GPU to save velocity data into a buffer on the GPU device, and copies that buffer back to the CPU host. After data copying is done, the output thread signals the main thread to compute the next timestep. At the same time, the output thread prepares the aggregated data to write to the disk while waiting for signal for the new generated velocity data. Therefore, the output task removes some non-computation activities from the main thread to make better use of the computing resources. Other potential tasks will be related to post-processing tools, visualization, analysis tools to gather statistics from the run, or interactive control tools.

3.5 Implementation of SGT Calculations

The SGT generation step is by far the most time-consuming processing step in the CyberShake workflow, accounting for approximately 90% of the CPU-hours. By adapting the GPU solver AWP for CyberShake, we are able to accelerate the SGT calculations (hereafter abbreviated AWP-SGT). The code supports 2D decomposition on GPUs, where each GPU is responsible for performing stress, velocity and SGT calculations within its own subgrid of the simulation volume. Two IO communication schemes are implemented for calculating SGTs. The first uses serial IO with the velocity mesh partitioned in advance. The second utilizes run-time partitioning inside the solver, using MPI-IO. This code has been extensively verified by comparing stress and strain outputs of earthquake sources to those from a reference model. Such verification is crucial during optimization and code updates.

3.5.1 Co-scheduling

When the SGT calculations are performed on GPUs, the CPUs on the same nodes are mostly idle except for handling IO and communications, which could be a potential waste of resources. To maximize our results using heterogeneous supercomputers, we have developed a runtime environment for co-scheduling across CPUs and GPUs. A CyberShake workflow consists of two phases: (1) two parallel AWP-SGT calculations, and (2) high-throughput reciprocity calculations with each rupture variation to produce seismograms and intensity measures of interest. These phases are described in more detail in Section 5.2. Co-scheduling enables us to perform both phases of a CyberShake calculation simultaneously on XK7 nodes, reducing our time-to-solution and making efficient use of all available computational resources. The following is a pseudo-code for multiple simultaneous parallel jobs:

```
aprun -n <#GPUs to use>  <AWP-SGT executable> <arguments> &
get the PID of the GPU job
cybershake_coscheduling.py:
    build all the cybershake input files
    divide up the nodes and work among a customizable number of jobs
    for each job:
        fork extract_sgt.py cores --> performs pre-processing and
        launches
        "aprun -n <cores per job> -N 15 -r 1 <cpu executable A>&"
        get PID of the CPU job
    while executable A jobs are running:
        check PIDs to see if job has completed
        if completed:launch
            "aprun -n <cores per job> -N 15 -r 1 <cpu executable
        B>&"
    while executable B jobs are running:
        check for completion
check for GPU job completion
```

To enable co-scheduling, we launch multiple MPI jobs on XK7 nodes via multiple calls to *aprun*, the ALPS utility to launch jobs on compute nodes from a Node Manager (MOM) node. We use core specialization when launching the child *aprun* calls to keep a core available for GPU data transfer and communication calls, as AWP-SGT uses MPI. Testing has shown that this approach results in little to no impact on either the GPU or CPU performance. To prevent overloading the MOM node with too many simultaneous *aprun* calls, we limit the number of child *aprun* calls to no more than 5.

The two executables, referred to here as "cpu executable A" and "cpu executable B", correspond to the two different codes which are executed as part of phase 2 of the CyberShake workflow. Executable A is an MPI executable which extracts and reformats the SGTs into multiple files, two for each of about 7000 ruptures, and executable B synthesizes synthetic seismograms and calculates peak acceleration values for each of about 410,000 earthquakes in an MPI master/worker environment. We invoke these executables separately to give us finer-grained control. We

obtain better performance by dividing the work across multiple jobs. This may be due to non-optimal load balancing which we plan to investigate and improve in the future.

Since calculating a pair of SGTs requires approximately 60 GPU hours, and the CyberShake post-processing requires about 1,000 CPU hours, the post-processing is able to complete on the 15 available CPUs per XK7 node while SGTs are calculated on the GPUs. We have successfully verified a CyberShake hazard curve using this co-scheduling approach, calculating SGTs on 50 GPUs while performing post-processing with 5 child jobs of 10 nodes each.

3.5.2 Hazard Curve Calculation

PSHA results are typically delivered as seismic hazard maps, or as site-specific seismic hazard curves. Probabilistic seismic hazard curves relate peak ground motion on the X-axis to probability of exceeding that level of ground motion on the Y-axis, for a site of interest. To verify AWP-SGT, we calculated a CyberShake hazard curve compared to the one generated using the CPU code. Both GPU and CPU results are completely overlapping as shown in Figure 8. Calculation of a hazard curve involves SGT time series data from over half a million locations in the volume, providing rigorous verification.

3.6 Verification

We performed a variety of tests to ensure that AWP produces results comparable in accuracy to other widely used and validated SCEC community codes running on HPC systems. We started with a wave propagation simulation of the magnitude-5.4 Chino Hills earthquake at frequencies up to 2.5 Hz using 128 Keeneland GPUs, with extended sources active for 2.5 seconds. The comparison of visualization of surface velocity verified that the GPU simulation results and original validated CPU results are identical over the tested 25,000 time steps [34].

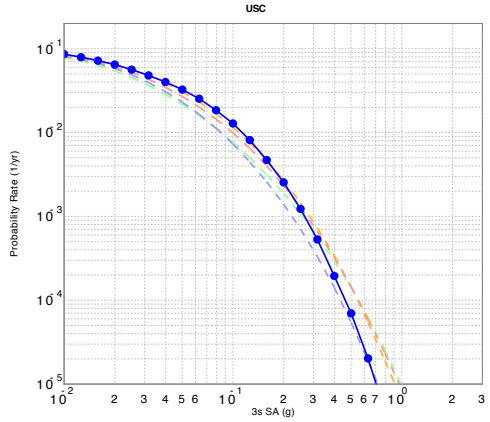

Figure 8: PSHA hazard curve calculated for the University of Southern California (USC) site. The horizontal axis represents ground motion at 3 seconds spectral acceleration, in terms of *g* (acceleration due to gravity). The vertical axis gives the probability of exceeding that level of ground motion. The blue line is the curve calculated using CyberShake with AWP-SGT, showing identical results compared to the CPU code. The dashed lines are hazard curves calculated using four common attenuation relationships which provide validation of the CyberShake methodology.

Comparing the velocities with negligible error is necessary, but not sufficient for the execution of accurate simulation of ground motions. Even small errors can accumulate over time if they are correlated or biased. We then examined seismograms generated using the SGT calculations. The results from the GPU code and reference model are nearly identical, with a maximum difference of 0.09%, in 1.2 billion mesh point volume for 20K timesteps.

4. PERFORMANCE ANALYSIS

We present the scaling results obtained on OLCF Titan, NCSA Blue Waters and Georgia Tech Keeneland.

AWP has undergone extensive fine-tuning on NVIDIA Fermi GPUs, but the team has had only limited time to analyze performance and optimize for NVIDIA Kepler GPUs, such as those in Titan and Blue Waters. We have observed that for small sub-domain sizes, accessing input arrays through the GPU's texture cache sped up the two primary compute kernels by a combined 1.9X. This speed-up is due to a reduction in global memory transactions. At larger sub-domain sizes, such as those used for the scaling results below (approximately 52 million grid points), the local data becomes too large for the texture cache, which negates the benefit of this change. We did see more modest gains by loading some, but not all, input arrays through the texture cache. In the future we intend to explore the use of per-SM shared memory to more selectively stage data arrays to achieve the same reduction in global memory transactions. This may have the additional benefit of reducing the register usage per thread and increasing occupancy.

4.1 Benchmark Machine Specifications

The OLCF Titan is a Cray XK7 supercomputer located at the Oak Ridge Leadership Computing Facility (OLCF), with a theoretical peak double-precision, floating point performance of more than 20 petaflops. Titan consists of 18,688 physical compute nodes, where each compute node is comprised of one 16-core 2.2GHz AMD Opteron™ 6274 (Interlagos) CPU, one NVIDIA Kepler GPU, and 32 GB of RAM. Two nodes share a Gemini™ high-speed interconnect router. Nodes within the compute partition are connected in a 3D torus [20]. The Blue Waters system is a Cray XE6/XK7 hybrid machine composed of AMD 6276 "Interlagos" processors (nominal clock speed of at least 2.3 GHz), NVIDIA K20X GPUs, and Cray Gemini interconnect [30]. The Keeneland Full Scale (KFS) system consists of a 264-node cluster based on HP SL250 servers. Each node has 32 GB of host memory, two Intel Sandy Bridge CPU's, three NVIDIA M2090 (Fermi) GPUs, and a Mellanox FDR InfiniBand interconnect. The total peak double precision performance is around 615 TFlops [13].

4.2 Scaling

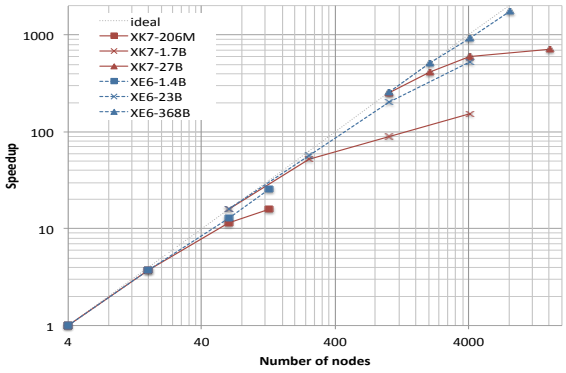

Figure 9: Strong scaling of AWP on Cray XE6/XK7 at ORNL and NCSA, with varied problem size from 206 million to 368 billion grid points.

The strong scaling benchmarks were performed on NCSA Blue Waters and OLCF Titan (Figure 9). The degradation in performance with the increase of the number of GPUs is expected, as we see the workload starvation of highly efficient number-crunching GPUs competing for the fixed total workload, plus the application bounded by communication overhead that arises from less compute work. This is quite different from the CPU code where we measured 85.6% of parallel efficiency for strong scaling between 1,024 and 8,192 XE6 nodes without topology tuning, and on earlier Cray system XT5 we observed super-linear speedup up to 223K Jaguar cores due to efficient cache utilization [7]. As the number of GPUs increases, so does the outer halo region to total sub-volume size ratio in proportion, making our application less effective in overlapping communication and computation. However, strong scaling is less important than weak scaling for our CyberShake calculations, in which tens of thousands of SGT simulations are involved.

In weak scaling benchmarks, each GPU carries out stencil calculations for a sub-domain with size 160 × 160 × 2048. This sub-domain size is also used for science simulations discussed later in Section 5. The total number of grid points in the domain becomes 160 × 160 × 2048 × N, where N is the number of GPUs used. Since we have 2D decomposition, the scaling is done in two dimensions of the computational domain (X and Y dimensions).

In the benchmark tests, linear speedup was observed on 90 Keeneland Initial Delivery System (KIDS) nodes equipped with 3 NVIDIA M2090 GPUs per node, where 10% of the peak performance was achieved. Figure 10 and Table 1 show the AWP code's scaling performance with 100% parallel efficiency for weak scaling from 16 up to 8192 Titan nodes. Linear weak scaling indicates that our careful design of communication model is highly efficient. Notable slowdown was observed in the case of 16,384 nodes, although AWP still achieves 93.5% parallel efficiency. Since the application performs only nearest-neighbor communications, we would expect continued linear scaling. The cause of this performance degradation is not yet fully understood, but in another related work we found a significant benefit in mapping application's virtual mesh topology to the allocated physical Blue Waters subnet topology. Topology tuning with Cray's *Topaware* tool, that is to match the virtual 3D Cartesian topology to an elongated physical subnet prism shape in the Cray XE6/XK7 torus, boosted 35% of strong scaling performance for fixed 45 billion mesh point size on a tested 4,096-node case on XE6 nodes.

4.3 Sustained Performance

We calculate the performance by measuring the average time spent on one time step after running a benchmark test for 2,000 time steps. The number of floating point operations counted in the code is based on 307 Flops per mesh point per time step. Initialization and output writing are excluded from this calculation. The IO time is negligible when iterations of hundreds of thousands of time steps are involved. We obtained a sustained performance estimate of 2.33 Pflops on 16,384 Titan GPUs. This was a 2,000 time-step benchmark run of a problem size of 20,480 × 20,480 × 2,048 or 859 billion mesh points.

Our main scientific findings using the code were obtained from a rough-fault simulation with a domain size 416 km × 208 km × 41 km with a spatial resolution of 20 meters at a maximum frequency resolution of 10-Hz, discretized into 443 billion mesh points. The size of this run is slightly larger than the record M8 San Andreas fault simulation [7]. The run took only 5 hours and 30 minutes to complete 170 seconds of simulation time using 16,640 Titan GPUs whereas M8 ran on approximately 220K CPU cores for 24 hours. We emphasize that the 10-Hz rough-fault simulation included 6.8 TB input and 170 GB output. To our best knowledge, this is the first sustained petaflop seismic production simulation to date, and a new record for earthquake simulation in terms of scale and scalability. These results are particularly remarkable considering that memory-bound stencil computations typically achieve a low fraction of theoretical peak performance.

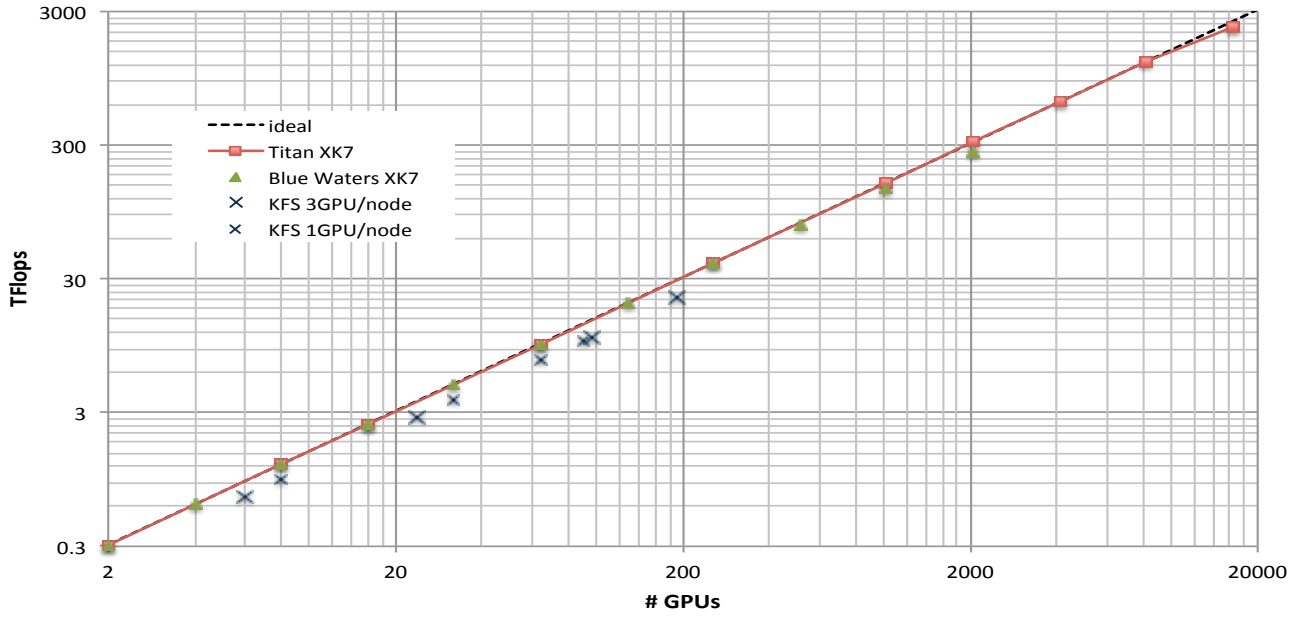

Figure 10: Weak scaling of AWP on OLCF Titan and NCSA Blue Waters, square/triangle/cross points are Flops performance on Titan/Blue Waters/KFS(Keeneland). A perfect linear speedup is observed between 16 and 8,192 nodes. A sustained 2.3 Pflop/s performance in single precision was recorded on 16,384 Titan nodes.

Table 1: Time-to-solutions and Parallel efficiency

XK7 Nodes used	Elements (Thousands)	Wall Clock Time	Parallel Efficiency
16 (4 × 4)	838,860	0.1085	100%
32 (4 × 8)	1,677,721	0.1084	100%
64 (8 × 8)	3,355,443	0.1085	100%
128 (8 × 16)	6,710,886	0.1085	100%
256 (16 × 16)	13,421,772	0.1085	100%
512 (16 × 32)	26,843,545	0.1085	100%
1024 (32 × 32)	53,687,091	0.1085	100%
2048 (32 × 64)	107,374,182	0.1084	100%
4096 (64 × 64)	214,748,364	0.1085	100%
8192 (64 × 128)	429,496,729	0.1085	100%
16384 (128 × 128)	858,993,459	0.1159	93.2%

Both the benchmark and rough fault runs produced remarkable scaling results. We also compared the performance against CPU systems. The benchmark results indicate that using GPU accelerators on Cray XK7 improves the performance by a factor of 5.2 compared to CPU-only usage of XK7 nodes. Furthermore the performance of XK7 exceeds XE6 by a factor of 2.5 when 512 nodes are in use. We expect our code's performance on XK7 nodes to improve further compared to XE6 as the number of nodes increases. This is because the CPU code suffers more from the increasing communication costs as it lacks effective overlap.

4.4 Time-to-solution and Performance-to-cost Analysis for CyberShake Calculations

One of the primary motivations of implementing AWP is to accelerate CyberShake calculations. We are planning to use CyberShake to calculate a California state-wide seismic hazard map with a maximum frequency of 1 Hz. When using the heavily optimized CPU code AWP-ODC, it is expected to require 662 million allocation hours to complete. Our AWP-SGT GPU code running on XK7 demonstrates a performance improvement of a factor of 3.7 compared to the CPU code running on XE6. Table 2 provides some detailed comparisons of calculating SGTs on XK7 versus XE6, based on the actual measurements of 0-0.5 Hz CyberShake production runs on Blue Waters XE6/XK7. We demonstrate the potential saving of 579 million allocation hours for total 5,000 sites required to generate the state-wide map, when using the accelerated (CPU+GPU) co-scheduling.

Table 2: CyberShake Strain Green Tensor Calculations

CyberShake	CPU[1] only	GPU[2] only	CPU+GPU[2]
XE6[1]/XK7[2] nodes	400	400	400
WCT[3] per site	10.36 hr	2.80 hr	2.80 hr
Total SUs charged[4]	662 M	168 M	168 M
Saved in Million SU[5]		495 M	**579 M**

1) XE6 node (dual AMD Interlagos); 2) XK7 (Interlagos+Kepler); 3) Wall clock time based on measurements on Cray XE6/XK7 at NCSA for two Strain Green Tensor calculations per site; 4) Based on total 5000 sites required for the generation of California state-wide seismic hazard map at a maximum frequency resolution of 1-Hz; 5) CPU+GPU saving includes 84 million XK7 Interlagos CPU hours without allocation charge for post-processing of seismogram extraction as co-scheduling, involving 6.2 million rupture variations calculations per site.

5. SCIENTIFIC RESULTS

We have applied these new AWP capabilities to obtain the first 10-Hz deterministic simulation on the *Titan* system and the first CyberShake hazard curve on the NCSA *Blue Waters* system.

5.1 Ground Motion Up To 10-Hz

High-frequency (>1 Hz) deterministic ground motion predictions are computation-based seismic hazard data products used as critical inputs to performance-based building design. The accuracy of the current, physics-based, ground motion simulations is limited by the small-scale complexity of the source and by high-frequency wave scattering in the crust. To investigate this problem, we have simulated high-frequency ground motions on a mesh comprising 443-billion (20,800 × 10,400 × 2,048) elements in a calculation that includes both small-scale fault geometry and media complexity. Specifically, we have computed ground motion synthetics using dynamic rupture propagation along a rough fault embedded in a 3D velocity structure with small-scale heterogeneities described by a statistical model.

We first carried out simulations of dynamic ruptures using a support operator method [26], in which the assumed fault roughness followed a self-similar fractal distribution with wavelength scales spanning three orders of magnitude, from ~10^2 m to ~10^5 m. We then used AWP to propagate the ground motions out to large distances from the fault in a characteristic 1D rock model with, and without, small-scale heterogeneities. The latter employed the moment-rate time histories from the dynamic rupture simulations as kinematic sources. Figure 11 shows snapshots of the rupture surface wave propagation for crustal models with and without the media heterogeneities. The fractal roughness is controlled by a Hurst number, which we set at 0.2, and the size of the heterogeneity by a standard deviation, which we set at 5%, as constrained by near-surface and borehole velocity data. Note how the wavefield in the bottom snapshot is scattered by the small-scale heterogeneities, which generates realistic high-frequency synthetics. A few seismograms are shown to compare models with and without the small-scale structure.

The simulation results show realistic features. The acceleration spectra from the simulation are nearly flat up to almost 10 Hz, in agreement with theoretical predictions. Moreover, the simulated response spectra compare favorably with spectra obtained from the empirical ground motion prediction equations (GMPEs) currently used by building engineers, which are calibrated to high-frequency recordings of earthquake ground motions.

5.2 Cybershake Hazard Model

Probabilistic seismic hazard models, including the CyberShake hazard model, estimate the probability that earthquake ground motions at a location of interest will exceed some intensity measure, such as peak ground acceleration, over a given time period. Results are delivered as hazard curves for a single site, and hazard maps for a region as illustrated in Figure 1.

To calculate a waveform-based seismic hazard estimate for a site of interest, we begin with UCERF [11] and generate multiple rupture realizations with differing hypocenter locations and slip distributions (sampled from an appropriate stochastic rupture model). A geo-referenced mesh of about 1.5 billion points (3,100 × 2,100 × 200) was constructed and populated with seismic wave velocity information from a SCEC Community Velocity Model. A body-force impulse is placed at the site of interest and the resulting 20K timestep simulation illuminates the volume, calculating SGTs. Seismic reciprocity is used to post-process the SGTs and obtain synthetic seismograms and peak intensity measures for each rupture variation [32]. These are combined with the UCERF rupture probabilities to produce probabilistic seismic hazard curves for the site using the OpenSHA hazard analysis code [12]. Figure 8 compares 3 s spectral acceleration AWP-SGT CyberShake hazard curves against standard curves derived from the reference GMPEs, computed for USC.

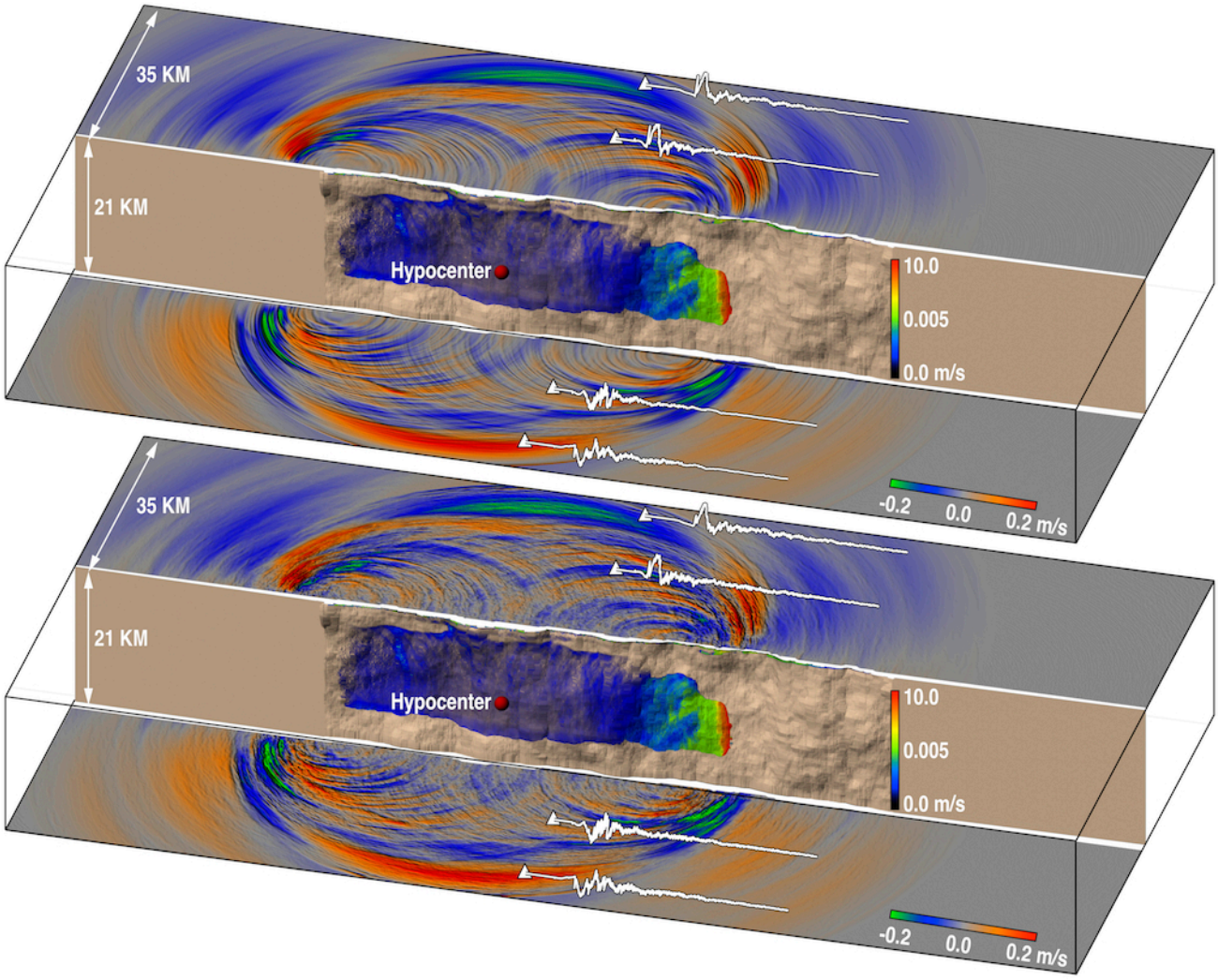

Figure 11: Snapshots of 10-Hz rupture propagation (slip rate) and surface wavefield (strike-parallel component) for a crustal model (top) without and (bottom) with a statistical model of small-scale heterogeneities. The displayed geometrical complexities on the fault were included in the rupture simulation. The associated synthetic strike-parallel component seismograms are superimposed as black traces on the surface at selected sites. The part of the crustal model located in front of the fault has been lowered for a better view. Note the strongly scattered wavefield in the bottom snapshot due to the small-scale heterogeneities.

A major computational challenge is how to increase the overall computational efficiency of the CyberShake workflow (Figure 12), which must combine the execution of the massively parallel SGT calculations with many loosely coupled post-processing jobs. We have successfully utilized workflow tools to manage the data and job dependencies [3]. Looking ahead, we plan to increase the frequency of the model from 0.5 Hz to 1.0 Hz, which will require simulations with eight times the mesh points and twice the timesteps.

Through an innovative co-scheduling approach, we have shown how CyberShake can make efficient use of GPUs and CPUs in heterogeneous systems. By running AWP-SGT on GPUs and doing high-throughput computations on the CPUs, we are able to run CyberShake workflows at a scale which now brings a 1-Hz California-wide CyberShake hazard model within reach.

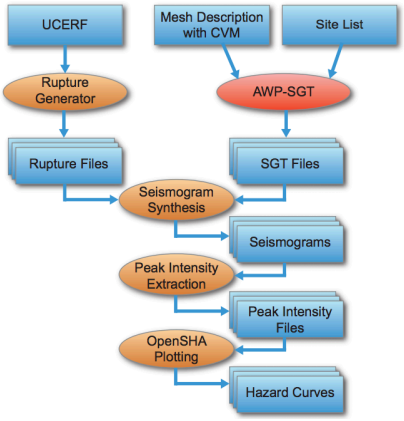

Figure 12: CyberShake workflow. Circles indicate computational modules and rectangles indicate files and databases.

6. CONCLUSIONS AND FUTURE WORK

We have re-designed the AWP-ODC code to accelerate wave propagation simulations on GPU-powered heterogeneous systems. An aggressive architecture-oriented optimization has maximized throughput and memory locality, providing much better performance in terms of speedup than our highly optimized CPU-based code. Algorithm-level communication reduction, effective overlap of communication and computation, and scalable IO have produced a GPU-based AWP code that achieves linear scalability and a sustained petaflops capability running seismic hazard research simulations. Although targeting GPU-based architectures, the basic concept of this paper – namely maximized throughput, memory locality, communication reduction, and scalable IO – is valid when porting to future many-core architectures.

AWP provides scientists, for the first time, with the ability to simulate ground motions from large fault ruptures to frequencies as high as 10 Hz in a physically realistic way. We have demonstrated this capability with simulations that incorporate both the fractal roughness of faults, which is thought to enhance the generation of high-frequency seismic waves, and the fractal heterogeneity of the crust, through which the waves are strongly scattered. The resulting ground motions compare favorably with leading GMPEs and provide guidance to further refine high-frequency simulations.

These results will change how synthetic seismograms are produced for use in earthquake engineering. Currently, the only way to compute synthetic seismograms across the full bandwidth of engineering interest (0.1-10 Hz) is to combine low-frequency deterministic simulations with high-frequency stochastic simulations [15, 18, 25]. The latter are obtained from *ad hoc* models that match the observed spectral content of the observations but do not satisfy the anelastic wave equations. The lack of a physics-based model makes it difficult to translate what is learned about the high-frequency behavior of one earthquake into forecasting the effects of future earthquakes. Our research shows how more realistic physics can be incorporated into future models.

We have also transformed the GPU-powered AWP to calculate SGTs. Our results show that AWP-SGT can serve as the main computational engine for CyberShake. The use of the AWP-SGT code is expected to save up to 495 million core-hours of computation for the proposed statewide CyberShake model.

The GPU-based AWP-SGT code will also provide highly scalable solutions for other problems of interest to SCEC as well as the wider scientific community, including full-3D waveform inversions to obtain more accurate velocity models for use in structural studies of the Earth across a range of geographic scales.

In the near future, we will refine our co-scheduling strategy for the CyberShake calculations to allow full utilization of both CPUs and GPUs on hybrid heterogeneous systems. Factor-of-three reductions in time-to-solution are anticipated, which will enable on-demand hazard curve calculations. We also plan to continue optimization of AWP on NVIDIA Kepler and develop resilience features. Finally, we are in the process of adding additional physics models to AWP-SGT simulations, incorporating more realistic media, different realizations of fault roughness, plasticity, and other features.

7. ACKNOWLEDGMENTS

We thank Jeffrey Vetter, Matthew Norman and Bob Fiedler of ORNL, Carl Ponder and Roy Kim of NVIDIA, Mitchel Horton of Georgia Tech, Sreeram Potluri and D. K. Panda of OSU, Didem Unat and Scott Baden of UCSD, for their major contributions of technical support; Jack Wells and Judith Hill of ORNL, Jay Alameda and Gregory Bauer of NCSA, Bruce Loftis of NICS for their resource support. The authors acknowledge the Office of Science of the U.S. Department of Energy (DOE) for providing HPC resources that have contributed to the research results reported within this paper through an Innovative and Novel Computational Impact on Theory and Experiment (INCITE) program allocation award. Computations were performed on Titan, which is part of the Oak Ridge Leadership Facility at the Oak Ridge National Laboratory which is supported by under DOE Contract No. DE-AC05-00OR22725. Compute resources used for this research are supported by XSEDE under NSF grant number OCI-1053575. Blue Waters computing resources was provided by NCSA. Research funding was provided through XSEDE's Extended Collaborative Support Service (ECSS) program, UCSD Graduate Program, Petascale Research in Earthquake System Science on Blue Waters PRAC (Petascale Computing Resource Allocation) under NSF award number OCI-0832698, SCEC's NSF Geoinformatics award: Community Computational Platforms for Developing Three-Dimensional Models of Earth Structure (EAR-1226343), and NSF Software Environment for Integrated Seismic Modeling (OCI-1148493). This research was supported by SCEC which is funded by NSF Cooperative Agreement EAR-0529922 and USGS Cooperative Agreement 07HQAG0008. The SCEC contribution number for this paper is 1753.

8. REFERENCES

[1] Bielak, J., Graves, R., Olsen, K. B., Taborda, R., Ramirez-Guzman, L., Day, S., Ely, G., Roten, D., Jordan, T., Maechling, P., Urbanic, J., Cui, Y. and Juve, G. 2010. The ShakeOut earthquake scenario: Verification of three simulation sets. *Geophysical Journal International.*, 180, 1 (Jan. 2010), 375-404.

[2] Blanch, J. O., Robertsson, J. O. and Symes, W. W. 1995. Modeling of a constant Q: methodology and algorithm for an efficient and optimally inexpensive viscoelastic technique. *Geophysics*, 60, 1, 176-184.

[3] Callaghan, S., Deelman, E., Gunter, D., Juve, G., Maechling, P., Brooks, C., Vahi, K., Milner, K., Graves, R., Field, E., Okaya, D. and Jordan, T. 2010. Scaling up workflow-based applications. *Journal of Computer and System Sciences*, 76, 6 (Sep. 2010), 428–446.

[4] Cerjan, C., Kosloff, D., Kosloff, R. and Reshef, M. 1985. A nonreflecting boundary condition for discrete acoustic and elastic wave equations. *Geophysics*, 50, 4, 705-708.

[5] Chen, P., Jordan, T. H. and Zhao, L. 2007. Full three-dimensional tomography: a comparison between the scattering- integral and adjoint-wavefield methods. *Geophysical Journal International.*, 170, 1, 175-181.

[6] Chen, P., Zhao, L. and Jordan, T. H. 2007, Full 3D tomography for the crustal structure of the Los Angeles region. *Bulletin of the Seismological Society of America*, 97, 4, 1094-1120.

[7] Cui, Y., Olsen, K. B., Jordan, T. H., Lee, K., Zhou, J., Small, P., Roten, D., Ely, G., Panda, D. K., Chourasia, A., Levesque, J., Day, S. M. and Maechling, P. 2010. Scalable earthquake simulation on petascale supercomputers. *Proc. of Int'l. Conf. for High Performance Computing, Networking,*

Storage and Analysis (SC'10, New Orleans, November 2010), 1-20.

[8] Dalguer, L. A. and Day, S. M. 2007. Staggered-grid split-node method for spontaneous rupture simulation. *Journal of Geophysical Research: Solid Earth*, 112, B02302. doi:10.1029/2006JB004467.

[9] Day, S. M. 1998. Efficient simulation of constant Q using coarse-grained memory variables. *Bulletin of the Seismological Society of America*, 88, 4, 1051-1062.

[10] Day, S. M. and Bradley, C. R. 2001. Memory-efficient simulation of anelastic wave propagation. *Bulletin of the Seismological Society of America*, 91, 3, 520-531.

[11] Field, E. H., Dawson, T. E., Felzer, K. R., Frankel, A. D., Gupta, V., Jordan, T. H., Parsons, T., Petersen, M. D., Stein, R. S., Weldon II, R. J. and Wills, C. J. 2009. Uniform california earthquake rupture forecast, version 2 (UCERF 2). *Bulletin of the Seismological Society of America*, vol. 99, no. 4 (Aug. 2009), 2053-2107.

[12] Field, E. H., Jordan, T. H. and Cornell, C. A. 2003. OpenSHA: A developing community-modeling environment for seismic hazard analysis. *Seismological Research Letters*, 74, 4, 406-419.

[13] Georgia Tech 2013. Keeneland User Guide. [Online]. https://www.xsede.org/gatech-keeneland.

[14] Graves, R. W. 1996. Simulating seismic wave propagation in 3D elastic media using staggered-grid finite differences. *Bulletin of the Seismological Society of America*, 86, 4, 1091-1106.

[15] Graves, R. W. and Pitarka, A. 2010. Broadband ground-motion simulation using a hybrid approach. *Bulletin of the Seismological Society of America*, 100, 5A, 2095-2123.

[16] Graves, R., Aagaard, B., Hudnut, K., Star, L., Stewart, J. and Jordan, T. H. 2008. Broadband simulations for Mw 7.8 southern San Andreas earthquakes: ground motion sensitivity to rupture speed. *Geophysical Research Letters*, 35, L22302 (Nov. 2008), doi:10.1029/2008GL035750, 1-5.

[17] Graves, R., Jordan, T. H., Callaghan, S. Deelman, E., Field, E., Juve, G., Kesselman, C., Maechling, P. Mehta, G., Milner, K., Okaya, D., Small, P. and Vahi, K. 2011. CyberShake: A physics-based seismic hazard model for southern california. *Pure and Applied Geophysics*, vol. 168, no. 3 (Mar. 2011), 367-381.

[18] Mai, P. M., Imperatori, W. and Olsen, K. B. 2010. Hybrid broadband ground motion simulations: combining long-period deterministic synthetics with high frequency multiple S-to-S back-scattering. *Bulletin of the Seismological Society of America*, 100, 5A, 2124-2142.

[19] National Research Council 2011. National earthquake resilience: Research, implementation, and outreach. *National Academies Press*, 198.

[20] Oak Ridge Leadership Computing Facility 2013. Titan User Guide. [Online]. https://www.olcf.ornl.gov/support/system-user-guides/titan-user-guide/.

[21] Olsen, K. B. 1994. Simulation of three-dimensional wave propagation in the Salt Lake basin. *University of Utah, Doctoral dissertation*.

[22] Olsen, K. B., Day, S. M., Minster, J. B., Cui, Y., Chourasia, A., Faerman, M., Moore, R., Maechling, P. and Jordan, T. H. 2006. Strong shaking in Los Angeles expected from southern San Andreas earthquake, *Geophysical Research Letters*, 33, 7 (Apr. 2006).

[23] Olsen, K. B., Day, S. M., Minster, J. B., Cui, Y., Chourasia, Okaya, D., Maechling, P. and Jordan, T. H. 2008. TeraShake2: spontaneous rupture simulations of mw 7.7 earthquakes on the southern San Andreas fault. *Bulletin of the Seismological Society of America*, vol. 98, no. 3, (Jun. 2008), 1162-1185.

[24] Porter, K., Hudnut, K., Perry, S., Reichle, M., Scawthorn, C. and Wein, A. 2011. Forward to the special issue on ShakeOut. *Earthquake Spectra*, 27, 2, 235-237.

[25] Schmedes, J., Archuleta, R. J. and Lavallée, D. 2010. Correlation of earthquake source parameters inferred from dynamic rupture simulations. *Journal of Geophysical Research: Solid Earth*, 115, B3.

[26] Shi, Z. and Day, S. M. 2013. Rupture dynamics and ground motion from 3-D rough-fault simulations. *Journal of Geophysical research*, 118, 1-20.

[27] Southern California Earthquake Center. 2013. ShakeOut. [Online]. http://shakeout.org.

[28] Taborda, R. and Bielak, J. 2013. Ground-motion simulation and validation of the 2008 Chino Hills. *Bulletin of the Seismological Society of America*, 103, 131-156.

[29] Tape, C., Liu, Q., Maggi, A. and Tromp, J. 2010. Seismic tomography of the southern California crust based on spectral-element and adjoint methods. *Geophysical Journal International.*, 180, 1, 433-462.

[30] University of Illinois NCSA 2013. Blue Waters System Overview. [Online]. https://bluewaters.ncsa.illinois.edu/user-guide.

[31] Wald, D. J. and Graves R.W. 2001. Resolution analysis of finite fault source inversion using one-and three-dimensional Green's functions: 2. Combining seismic and geodetic data. *Journal of Geophysical Research*, 106.

[32] Zhao, L., Chen, P. and. Jordan, T. H. 2006. Strain Green's tensors, reciprocity, and their applications to seismic source and structure studies. *Bulletin of the Seismological Society of America*, 96, 5, 1753-1763.

[33] Zhou, J., Unat, D., Choi, D., Guest, C. an Cui, Y. 2012. Hands-on performance tuning of 3D finite difference earthquake simulation on GPU fermi chipset. In *Proc. of Int'l. Conf. on Computational Science* (ICCS'12, Omaha, Nebraska, June 4-6, 2012), 9, 976-985.

[34] Zhou, J., Cui, Y., Poyraz, E., Choi, D. and Guest, C. 2013. Multi-GPU implementation of a 3D finite difference time domain earthquake code on heterogeneous supercomputers. In *Proc. of Int'l. Conf. on Computational Science* (ICCS'13, Barcelona, June 5-7, 2013), 18, 1255-1264.